

Index Optimization by String Vector based KNN Variants

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the string vector based KNN variants, and applied them to the index optimization. The initial KNN version was previously modified into the string vector-based version as an approach to the index optimization which was mapped into a classification task. In this research, we cover the three KNN variants in the case of the numerical vector-based versions: one where the selected nearest neighbors are discriminated by their similarities, one where the attributes are discriminated by their correlations with the categories, and one where the training examples are discriminated by their credits. In this research, we modify the three KNN variants into the string vector-based version, as well as the initial KNN version. This research is aimed to improve the classification performance by modifying the KNN variants so.

I. INTRODUCTION

The index optimization is defined as the process of filling the text index with relevant words for maximizing the information retrieval performance. The index optimization was mapped into a word classification where each word is classified into expansion, inclusion, or removal. The index optimization should be treated differently from the topic-based word classification where a word is classified into one or some among the predefined topics, in that it is a domain dependent task where a same entity may be classified differently, depending on the given domain. In the index optimization system, a text is indexed into a list of words, each word is classified into one of the three categories, and the actions are taken differently to individual words by their labels. The actions to the classified words are addition of associative words from external sources to one which are classified with expansion as the index expansion, inclusion of words by themselves to ones which are classified with inclusion, and exclusion of words to ones which are classified with removal.

This research is motivated directly by the successful results from applying the string vector based KNN algorithm to the index optimization. Indirect motivations for this research are its successful application to other tasks, such as the topic-based word categorization and the keyword extraction. The similarity metric between two string vectors was defined, and the KNN algorithm was modified into the string vector-based version which processes directly string vectors. Its better performance was empirically validated in

the index optimization, by comparing it with the traditional KNN algorithm. In this research, we modify the variants into the string vector-based versions, as well as the KNN algorithm.

The idea of this research is to modify the three KNN variants into the string vector-based versions and apply them to the index optimization which is mapped into a classification task. The first variant weights nearest neighbors by their similarities with or their distances from a novice example in voting their labels. The second variant weights attributes by their correlations with target outputs in computing similarities of a novice example with the training examples. The last variant weights the training examples by their qualities in computing their similarities with a novice item or voting nearest neighbors. This research is intended to improve their performances by modifying the three KNN variants into the string vector-based versions in the index optimization.

Let us mention some benefits from this research. The problems which are caused by encoding words into numerical vectors are solved by proposing string vectors as alternative word representations. The reliability is upgraded by substituting the KNN variants which discriminate attributes, training examples, and nearest neighbors for the KNN algorithm. The classification performance is improved by modifying the KNN variants into the string vector-based versions. More reliable approaches are provided for implementing the index optimization system by this research.

Let us mention the organization of this paper. In Section II, we explore the previous works which are relevant to this study. In Section III, we describe in detail what is proposed in this research. In Section IV, we mention the significances of this research and the remaining tasks. This paper is composed of the four sections.

II. PREVIOUS WORKS

This section is concerned with the previous works which are relevant to this research. It is intended to expand the style of modifying the KNN algorithm to its three variants as the approaches to index optimization. The directions of exploring previous works are derivation of KNN variants, modification of the standard KNN algorithm into its string vector-based version as an approach to the task, and the

previous case of encoding a text into a string vector. The distinguishable points of this research are derivation of three KNN variants from the standard version, modification of them into their string vector-based versions, and application of them to the index optimization. This article is intended to explore previous works for providing the background for this research.

Let us explore the previous works on KNN variants. In 2001, the KNN variant where nearest neighbors are discriminated by their distances from or similarities as a novice item was applied to the text classification by Han et al. [1]. In 2008, MKNN (Modified K Nearest Neighbor) the KNN variants where training examples are discriminated by their validness, was proposed by Parvin et al. [2]. In 2018, the KNN variant where the attributes are discriminated by their correlations were proposed by Nababan and Sitompul [10]. However, this research uses different specific metrics for discriminating attributes, nearest neighbors, and training examples.

Let us explore the previous works on applying the modified version of KNN algorithm to the index optimization. In 2016, it was initially proposed that the string vector-based version of KNN algorithm should be applied to the word categorization as a position paper by Jo [6]. In 2018, the modified KNN algorithm was validated in the task by Jo [8]. The journal article which describes in detail the modified KNN algorithm and its application to the index optimization is finally prepared for its publication by Jo [11]. This research is based on the three previous works.

Let us explore the previous works on the neural networks, NTC (Neural Text Categorizer), where encoding a text into a string vector was initially proposed. In 2010, it was initially invented as an approach to the text classification by Jo [3]. In 2015, it was applied to the topic identification of Arabic texts by Abainia [5]. In 2018, it was evaluated as the most practical neural networks by Lakshmi and Rao [9]. In 2010, the NTSO (Neural Text Self Organizer) was also invented as an approach to the text clustering by Jo [4].

Let us mention the differences of this research from the previous works which were explored above. The KNN variants in the previous works are numerical vector-based versions, and they will be modified into string vector-based versions in this research. As the expansion of [11], we modify and adopt the KNN variants for implementing the index optimization system. Machine learning algorithms will be innovated by inventing various ones which process directly string vectors as their input. The modified KNN variants will be described in detail in Section III.

III. PROPOSED WORK

This section is concerned with what is proposed in this research. In Section III-A, we describe the process of encoding a word into a string vector and the proposed similarity metric. In Section III-B, we describe the standard version

of KNN algorithm where the proposed similarity metric is adopted. In Section III-C, we derive the three variants from the standard version. In Section III-D, we present the system architecture of the index optimization system.

A. Similarity Metric

This section is concerned with the string vector as the representation of a word and the similarity between string vectors. The string vector is defined as a finite ordered set of strings; in the vector, the numerical values are replaced by the strings. It is required to define a similarity matrix for computing the similarity between elements. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a word into a string vector and the process of computing the similarity between string vectors.

The process of encoding words into string vectors is illustrated in Figure 1. In the word-text association, a text collection is constructed by gathering texts, and each word is associated with its own texts based on their information. The string vector features are defined as symbolic forms manually in the feature definition. In the feature value assignment, the string vector which represents a word is filled with text identifiers which correspond to the features. Refer to [7] for studying the encoding process in more detail.

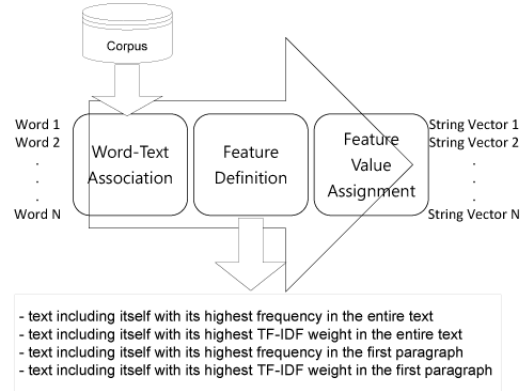


Figure 1. Process of Encoding Words into String Vectors

The similarity matrix which is the basis for computing the similarity between string vectors is illustrated in figure 00. In the matrix, each row and each column correspond to text identifiers, and each element is the similarity between texts, d_i and d_j , as shown in equation (1),

$$s_{ij} = \text{sim}(d_i, d_j) \quad (1)$$

The similarity between two texts, d_i and d_j , is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \quad (2)$$

where D_i is the set of words in the text, d_i , and D_j is the set of words in the text, d_j . The value of similarity between texts, d_i and d_j , is always given as a normalized value between zero and one, as shown in equation (3)

$$0 \leq \text{sim}(d_i, d_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent words as a reference table.

$$\begin{bmatrix} s_{11} & s_{12} & \dots & s_{1N} \\ s_{21} & s_{22} & \dots & s_{2N} \\ \dots & \dots & \dots & \dots \\ s_{N1} & s_{N2} & \dots & s_{NN} \end{bmatrix} \quad \text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$$

$$0 \leq \text{sim}(d_i, d_j) \leq 1.0$$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between words. The two string vectors which represent words are notated by $\text{str}_1 = [d_{11} \ d_{12} \ \dots \ d_{1d}]$ and $\text{str}_2 = [d_{21} \ d_{22} \ \dots \ d_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(d_{1i}, d_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(d_{1i}, d_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding words into string vectors and computing the similarity between words. A word is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between string vectors. The similarity is computed by equation (4). In the future research, we consider representing a word into multiple string vectors.

B. Initial Version of String Vector based KNN Algorithm

This section is concerned with the initial version of string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [7]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section

is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\text{str}_1, \text{str}_2, \dots, \text{str}_N\}$. A novice word is encoded into a string vector notated by str . Its similarities with the string vectors which represent the sample words are computed by equation (4), as $\text{sim}(\text{str}, \text{str}_1), \text{sim}(\text{str}, \text{str}_2), \dots, \text{sim}(\text{str}, \text{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

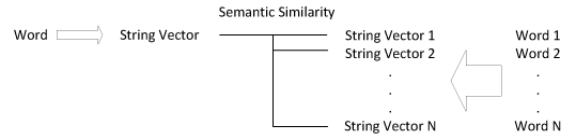


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, \text{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\text{str}_1^{\text{near}}, \text{str}_2^{\text{near}}, \dots, \text{str}_k^{\text{near}}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

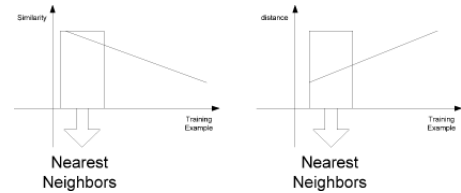


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\text{str}_i^{\text{near}})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$.

The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$\text{label}(\mathbf{str}) = \underset{j=1}{\operatorname{argmax}}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the initial version of KNN algorithm from which we derive some variants. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. KNN Variants

This section is concerned with the variants which are derived from the KNN algorithm which was covered in Section III-B. The difference from the traditional version is to adopt similarity metric between string vectors for computing the similarity between a novice item and a training example. In this section, we derive the three variants by discriminating nearest neighbors, attributes, or training examples. The three variants of KNN algorithm are adopted for implementing the index optimization system. This section is intended to describe the three variants.

Let us mention the KNN variant which is derived by discriminating the nearest neighbors. A set of nearest neighbors is notated by $Nr = \{\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}\}$, and its elements are assumed as $\text{sim}(\mathbf{str}, \mathbf{str}_1^{near}) \geq \text{sim}(\mathbf{str}, \mathbf{str}_2^{near}) \geq \dots \geq \text{sim}(\mathbf{str}, \mathbf{str}_k^{near})$. The weights, $w_1, w_2, \dots, w_k$, are assigned to the nearest neighbors, $\mathbf{str}_1^{near}, \mathbf{str}_2^{near}, \dots, \mathbf{str}_k^{near}$, following, $w_1 \geq w_2 \geq \dots \geq w_k$, and the total weights are computed for the category, c_j by equation (9),

$$CS(Nr, c_j) = \sum_{\mathbf{str}_i \in Nr} w_i \quad (9)$$

The label of the novice item, $\mathbf{str}$, is decided by equation (10),

$$\text{label}(\mathbf{str}) = \underset{j=1}{\operatorname{argmax}}^m CS(Nr, c_j) \quad (10)$$

There are two ways of assigning weights to the nearest neighbors: exponentially decreasing way and arithmetic decreasing way as shown in Figure 6.

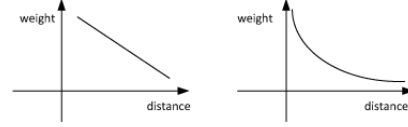


Figure 6. Discrimination over Nearest Neighbors

Let us mention the second KNN variant where the attributes are discriminated for computing the similarity between a novice item and a training example as shown in Figure 7. If the number of training examples is N , and the dimension of numerical vector representing a training example is d , the attributes are notated as a set, $A = \{A_1, A_2, \dots, A_d\}$, and each attribute is viewed as a set of N values $A_i = \{str_{1i}, str_{2i}, \dots, str_{Ni}\}$. The occurrences of the string value, str_{ij} , in the category, c_k , is notated by $f(str_{ij}|c_k)$. The influence of the attribute, A_i , on the category, c_k , is computed by equation (11),

$$I(A_i, c_k) = \frac{1}{N} \sum_{j=1}^N f(str_{ji}, c_k), \quad (11)$$

and the weight of the attribute, A_i , is computed by equation (12),

$$w_i = \max_{k=1}^m I(A_i, c_k). \quad (12)$$

The similarity between the novice input, $\mathbf{str}$ and the training example, $\mathbf{str}_i$, is computed by equation (13),

$$\text{sim}(\mathbf{str}_i, \mathbf{str}) = \frac{1}{d} \sum_{j=1}^d w_i \text{sim}(str_{ij}, str_j) \quad (13)$$

In this variant, the equation (4) is replaced by equation (13) for computing the similarity between a novice item and a training example.

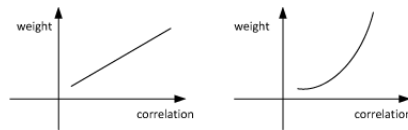


Figure 7. Discrimination over Attributes

Let us mention the third KNN variant where the training examples are discriminated for computing the similarity between a novice item and a training example and/or voting the nearest neighbors for deciding the label as shown in Figure 8. The similarity threshold is used as the parameter, and for each training example, other training examples are taken within the threshold from it as its nearest neighbors. The number of nearest neighbors and the number of training examples which are labeled identically to the training example, $\mathbf{str}_i$, are notated respectively by r_i and r_i^- , and the weight is computed by equation (14),

$$w_i = \frac{r_i^-}{r_i} \quad (14)$$

The similarity between the novice item, $\mathbf{str}$, and the training example, $\mathbf{str}_i$ is computed by equation (15),

$$\text{sim}(\mathbf{str}_i, \mathbf{str}) = \frac{w_i}{d} \sum_{j=1}^d \text{sim}(\text{str}_{ij}, \text{str}_j) \quad (15)$$

and the total weight is computed for the category, c_i , by equation (16), in voting,

$$CS(Nr, c_j) = \sum_{\mathbf{str}_i \in Nr} w_i. \quad (16)$$

Equation (15) and (16) are used respectively for computing the similarity between a novice item and a training example and voting the nearest neighbors for each category.

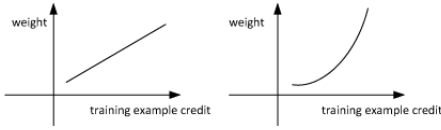


Figure 8. Discrimination over Training Examples

Let us make some remarks on the three variants of KNN algorithm which are proposed as the approaches to the index optimization. In the first variant, the nearest neighbors are discriminated for voting their labels. In the second variant, the attributes are discriminated for computing the similarity between a novice input and a training example. In the third variant, the training examples are discriminated for voting the labels of the nearest neighbors and/or computing the similarity between string vectors. We may consider the trainable KNN algorithm which optimizes the weights of the nearest neighbors, the attributes, and the training examples for minimizing the training error.

D. System Architecture

This section is concerned with the architecture and the execution flow of the index optimization system. In Section III-C, we described the three KNN variants which are adopted for implementing the index optimization system. The initial version of KNN algorithm which is mentioned in Section III-B is replaced by its variants in the architecture of the index optimization system which was previously proposed. The process of executing the system is to encode a word into a string vector and classify it into one of the three categories. This section is intended to describe the index optimization system which is implemented in this study.

The process of collecting the same words and encoding them into string vectors for implementing the index optimization system is illustrated in Figure 9. The index optimization is interpreted into a domain dependent classification; even a same word is classified differently, depending on the domain. For each domain, an independent classification task is defined. The several domains are decided, and the words which are labeled with one of the three categories

exclusively in each domain. It is required to present the domain from a text which is given as the input for doing this task.

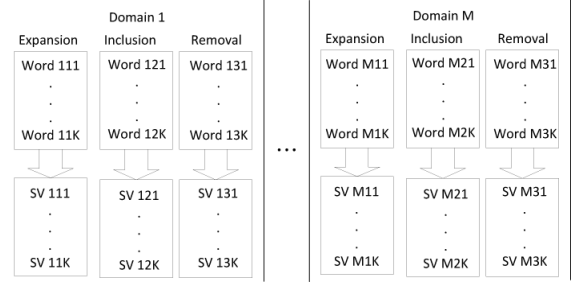


Figure 9. Preparation of Training Examples

The entire system architecture of the index optimization which is implemented in this research is illustrated in Figure 10. Words which are labeled with one of the three categories are collected as the sample words and are encoded into string vectors in the encoding module. The similarities of each word which is indexed from a text are computed in the similarity computation module. For each word in the text, its nearest neighbors are selected, and its category is decided in the voting module. The difference from the system architecture which is proposed in [8] is to replace the initial version of KNN algorithm by its variants.

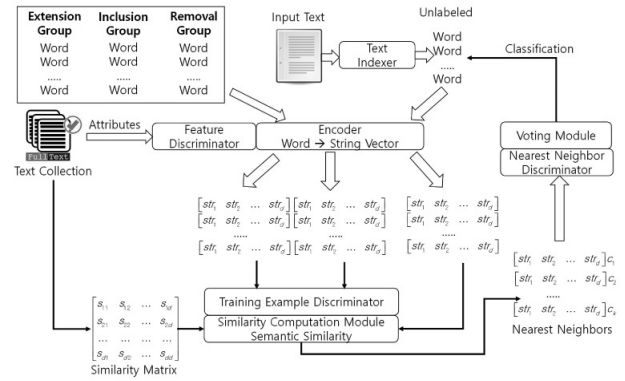


Figure 10. System Architecture

The execution process of the index optimization system is illustrated in Figure 11. Within a domain, texts are collected, sample words are prepared by labeling them with one of the three categories, and they are encoded into string vectors. A text is indexed into a list of words, and they are also encoded into string vectors. The words in the input text are classified into one of the three KNN variants. The words in the removal group are excluded, ones in the inclusion group are included in the index, and to ones in the expansion group, their associated words are added from external sources.

Let us make some remarks on the proposed system which is illustrated in Figure 10 as its architecture. The M domains

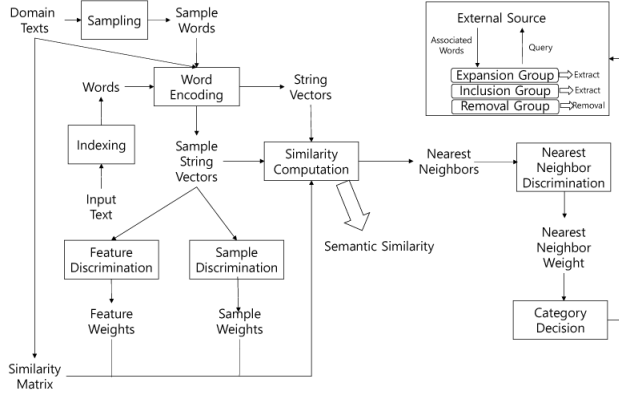


Figure 11. System Flow

are decided in advance, sample words each of which is labeled with one of the three categories are prepared in each domain, and they are encoded into string vectors. The words which are indexed from a text are classified by one of the three KNN variants which are described in Section III-C. The difference from the previous version of index optimization system is to provide the selection of the nearest neighbor discrimination, the attribute discrimination, and the training example discrimination. We expect the better performance by replacing the initial version by its variants.

IV. CONCLUSION

Let us mention the significances of this research as the conclusion. We encode words into string vectors and apply the similarity metric among them to the KNN variants as well as the standard version. We derive the three variants from the standard version by discriminating the nearest neighbors, the attributes, and the training examples. We design the index optimization system by adopt the KNN variants with the proposed similarity metric. In the next research, we will implement the index optimization system as a real system.

Let us mention the remaining tasks as the further discussions on this research. It is necessary to improve the speed of computing the similarity between string vectors in the implementation level. Other machine learning algorithms such as the Naïve Bayes and the SVM (Support Vector Machine) are modified into the string vector-based versions. The proposed versions of KNN variants are applied to other tasks such as text classification and text summarization. The index optimization system should be implemented by adopting the proposed approaches as a real program.

REFERENCES

- [1] E.H. Han, G. Karypis and V. Kumar, "Text Categorization Using Weight Adjusted k-Nearest Neighbour Classification" pp53-64, in *Advances in Knowledge Discovery and Data Mining. PAKDD 2001*, Berlin, Heidelberg:Springer, 2035, 2001.
- [2] H. Parvin, A. Hosein, and M.B. Behrouz, "MKNN: Modified k-nearest neighbor" *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 1. Newswood Limited, 2008.
- [3] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", 83-96, *International Journal of Information Studies*, 2, 2, 2010.
- [4] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", 31-43, *Journal of Network Technology*, 1, 1, 2010.
- [5] K. Abainia, S. Ouamour, and H. Sayoud. "Neural Text Categorizer for topic identification of noisy Arabic Texts." 1-8, *The Proceedings of IEEE/ACS 12th International Conference of Computer Systems and Applications*, 2015.
- [6] T. Jo, "String Vector based KNN for Index Optimization", 31-36, *The Proceedings of 3rd International Conference on Advances in Big Data Analysis*, 2016.
- [7] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, *Journal of Intelligent and Fuzzy Systems*, 35, 2018.
- [8] T. Jo, "String Vector based Version of K Nearest Neighbor for Index Optimization in Current Affairs", 47-50, *The Proceedings of International Conference on Applied Cognitive Computing*, 2018.
- [9] P. P. Lakshmi and D. R. Rao, "Text classification using artificial neural networks", 603-606, *International Journal of Engineering & Technology*, 7, 1, 2018.
- [10] A.A. Nababan and O. S. Sitompul, "Attribute weighting based K-nearest neighbor using Gain Ratio", *Journal of Physics: Conference Series*, 1007, 1, 2018.
- [11] T. Jo, "Specializing K Nearest Neighbor into String Vector based Version using String Vector Operation in Index Optimization", in *Preparation*, 2023.